2009

Content-based addressing in hierarchical distributed hash tables

Joseph Srebro

R·I·T

Content-Based Addressing in Hierarchical Distributed Hash Tables

by

Joseph Mark Srebro

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Science

Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY
October 2009

Chair Approval:

Minseok Kwon *Chair*	10/14/2009
Leon Reznik *Reader*	10/14/2009
Joe Geigel *Observer*	10/14/2009

Content-Based Addressing in Hierarchical Distributed Hash Tables

Joseph Mark Srebro (jms8200@rit.edu)

Abstract— **Peer-to-peer networks have drawn their strength from their ability to operate functionally without the use of a central agent. In recent years the development of the structured peer-to-peer network has further increased the distributed nature of p2p systems. These networks take advantage of an underlying distributed data structure, a common one is the distributed hash table (DHT). These peers use this structure to act as equals in a network, sharing the same responsibilities of maintaining and contributing. But herein lays the problem, not all peers are equal in terms of resources and power. And with no central agent to monitor and balance load , the heterogeneous nature of peers can cause many distribution or bottleneck issues on the network and peer levels. This is due to the way in which addresses are allocated in these DHTs. Often this function is carried out by a consistent hashing function. These functions although powerful in their simplicity and effectiveness are the stem of a crucial flaw. This flaw causes the random nature in which addresses are assigned both when considering peer identification and allocating resource ownership. This work proposes a solution to mitigate the random nature of address assignment in DHTs, leveraging two methodologies called hierarchical DHTs and content based addressing. Combining these methods would enable peers to work in cooperative groups of like interested peers in order to dynamically share the load between group members. Group formation and utilization relies on the actual resources a peer willingly shares and is able to contribute rather than a function of the random hash employed by traditional DHT p2p structures.**

I. INTRODUCTION

Peer-to-peer (p2p) networks have been the topic of several research works for the past few years. This quick adaptation can be attributed to their highly flexible and powerful nature. This allows them to be easily and beneficially applicable to a wide array of problems. These problems range from file sharing applications like Napster [14] to multicast message subscription services like Scribe [15]. In recent years the exploration of structured p2p networks has lead to the development of several powerful protocols. These protocols empowered by the use of distributed hash tables (DHTs) were able to achieve high scalability and generally quick response times of O(log n), based on the number of peers in the network. [1, 2].

The goal of this paper is to describe the importance of peer-to-peer load balancing as it applies to structured networks. Specifically proposed is a protocol using a methodology called content based addressing to distribute load fairly between peers. The problem of load balancing has been approached before by previous works such as replication strategies as well as address management protocols, time will be taken to show how the protocol proposed is fundamentally different from these approaches. In addition direct comparison will be done between these strategies using a network simulator to discern the benefits of using Content Based DHT (CBDHT) over competing strategies. The metrics of scalability and efficiency will be on the forefront of the discussion on results.

The rest of this paper will be organized as follows; *Section II* will discuss background information on the Chord protocol as well as related works done on Chord. *Section III* will comprise of discussing the shortcomings of the DHT architecture in terms of load balancing. This will lay down the motivation for investigating a load-balancing algorithm for DHTs. *Section IV* will outline the proposed algorithm for problems highlighted in *Section III*. *Section V* will contain information on the evaluation of the work. Metrics for comparison will be given here as will as the specification of simulators and test environments. This section will also serve to describe this particular implementation of CBDHT and quantify its general behavior. The results against baseline Chord and other load balancing techniques will be shown in *Section VI*. *Section VII* and will lay the grounds for future work and present a concise conclusion on this work.

II. RELATED WORK

One of the most fundamental examples of a structured p2p network is Chord. This thesis uses Chord as the underlying DHT protocol to implement the proposed load balancing scheme CBDHT over. Although choosing Chord in this particular implementation does not limit ideas to a singular protocol, as this scheme can be fully extensible to other DHT based routing protocols.

Developed at MIT, Chord was selected for several reasons. Chord at its purest is a very basic structure. It initially does not have any load balancing algorithms built into the protocol. This offers a viable test platform to implement the CBDHT load balancing algorithms without the fear of interference at the protocol level. Chord is also well established in structured p2p research. Thus many previous load-balancing algorithms have used Chord as their underlying algorithm as well, offering several chances to compare results within a common protocol base. This allows the proposed algorithm to show a clear benefit without having to make assumptions about performance metrics between other base protocols.

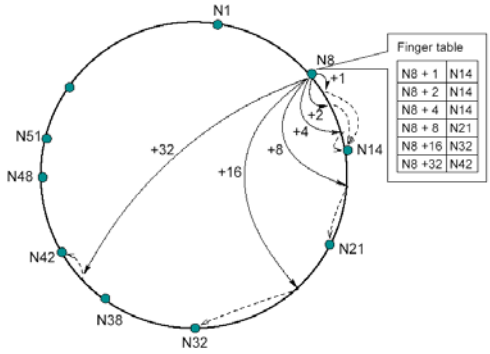

Figure 2.1 Shows a visual example of a Chord hash ring displaying the finger table of Node 8 (N8). [1]

As stated previously Chord is a DHT, which means it uses a consistent hashing function as the addressing function for both peers and the files they contain. These hashing functions take the place of filenames and IP addresses. It can be shown to route messages in O(log n) hops where n is the number of peers in the network. This is accomplished by the way each host stores the links to other hosts in the network. There are three types of links each node stores and maintains locally, a neighbor table, a successor table, and a finger table. Each table serves a specific purpose to the node in the Chord protocol. The two crucial tables are the successor and the finger tables; these tables are the heart of the Chord algorithm. The successor table is a simple list of links to nodes that lay directly ahead of the current node. For example in *Figure 2.1* N14 and N21 are good candidates to be in N8's successor table. The purpose of these nodes is to ensure that messages can be routed without error to the destination. But this type of chain routing at worst case operates in O(n) hops where n is the number of nodes. This occurs because each node is only

able to jump to nodes directly in front of it and in the worst case each node must perform this operation on its direct successor [1].

This is improved significantly by the use of the finger table. This table contains nodes that are a sizeable distance away from the successor peers, with the maximum hop being halfway around the address ring. In *Figure 2.1* N8's finger table is shown, it contains six entries representing several portions of the ring. It is important to note here that Chord uses a clockwise motion to resolve assignment. So while there is no direct node at the N8+1 location the protocol lists the next available node in this case N14. This allows a node to be listed several times in a finger table thus N14 has three entries in this example. So when routing a node will search its finger table for the address containing the closest matching address then forward the request onto this node. This is process is recursively carried out until the request reaches the destination. This process can be proven to operate in a respectable O(log n) hops where n is the number of nodes in the network [1]. File responsibility is distributed to each node in a similar way. Like message routing, file addressing is based upon a consistent hashing function that usually is a simple hash of the filename. Much like the construction of a finger table, responsibility of a file is handled in a clockwise motion around the ring. A node is responsible for a file if it is the direct successor to the file's hash. For example above N8 is responsible for all files in the address space between N1 and N8. In the event of a leave or new join, files will be redistributed based upon where the new node is placed based on its hash.

But as it is clear to see the benefits a distributed p2p network like Chord can offer. This distributive behavior can also be the source of many weaknesses. The architecture of these networks have no central server to manage bandwidth capabilities or detect overloads in arbitrary nodes. Thus extensive work has been done in the realm of balancing the load on these systems [3,4,5,6,7,8]. Although some benefits have been gained from the use of the algorithms described later in *Section II* each fails to provide coverage for the two fundamental load balancing problems in the p2p realm: address space misbalancing and the hotspot problem described further in *Section III*. This algorithm proposes the use of hierarchical DHTs coupled with an addressing technique that is more representative of the current peer to combat these two load balancing issues.

There has been much work done in the confronting these load balancing issues. These works have focused on two main approaches that focus on solving a particular issue. They are referred to as address space balancing (work stealing/virtual servers) and replication (file copying/caching).

Address space balancing works on the theory that since files are represented as space on the hash ring, that if the ring is able to be balanced evenly among the nodes that this will also balance the load among those nodes evenly as well. [5] Also in virtual server techniques this is taken a step further where peers represent themselves as nodes in several places on the hash ring in order take more of a fair share of the file distribution. [3].

This brings us to the second main idea in solving the load-balancing problem in structured p2p systems. This method is referred to as replication. This idea attempts to alleviate the hotspot problem by producing copies of popular files in strategic locations around the network often along common routing paths. Creating these replications enables nodes to serve as intermediate distributors of a particular popular file. This in turn spreads the network load between members of this replication group and even extends to distributing routing load in some cases [8].

Figure 2.2 is a simple illustration of an example of how a replication chain is organized. In this network there is a file named Popular.file, this file has a higher than average amount of requests for it. This file also originally resides solely under Node D's responsibility. After noticing the heightened amount of requests for Popular.file the peers A, B, and C send out requests of their own for a copy of Popular.file. After retrieving the file they can now serve as an intermediary distributor. This means there is a chance if a request reaches one of these peers for

Popular.file, they will then serve this file directly instead of passing the request along the chain to its original destination.

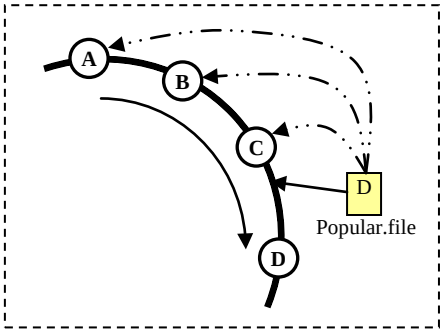

Figure 2.2 contains an example of a possible replication scheme. In replication copies of a popular file are passed onto nodes in possible routing paths (in direction of the arrow) to the node responsible. It is the hope that these nodes will serve as intermediate distributors of the file on subsequent requests.

The downside of this approach is the overhead of moving the replications onto nodes. Often times these nodes are not interested in the file thus their storage is useless to the peer that replicates it. These nodes also typically reside in direct routing paths to the original file of interest in order to service these replicates effectively. Proper placements of these files become difficult and often several copies are required for replications to become effective for each file [16]. This means that overhead ramps up linearly on a per file basis. This becomes increasingly problematic in the unlucky event several hotspots develop along the same portion of the ring. This causes those nodes in this portion of the ring to handle the majority of requests as well as the overhead of replicates now.

III. Problem Description

The problem of load balancing structured peer-to-peer networks is the crucial problem of this work. It focuses particularly on the problem of maintaining optimum fair performance in p2p systems. While some may argue that scalability should be the only major concern with these peer systems, fairness should also play a pivotal role. It is evident in heterogeneous environments like the Internet where machines can vary from 300Mhz dated systems behind 56k modems to multiprocessor supercomputers behind OC-3 lines that performance of individual nodes is important. Nor does this mean that these issues are mutually exclusive as poor resource management of a weak node could quite possibly limit the scalability of a system in real-time. Also it should not be ignored that files, similar to peers, are heterogeneous in nature. Of significant importance are the varying degrees of popularity these files can attain, as this popularity has direct correlation to the number of requests received for a file. There are two types of problems that can occur under these heterogonous conditions, Address space imbalance and the hotspot problem. Both problems however stem from the single crucial source this is its consistent hashing function, which used to drive these powerful protocols, is at heart random in nature.

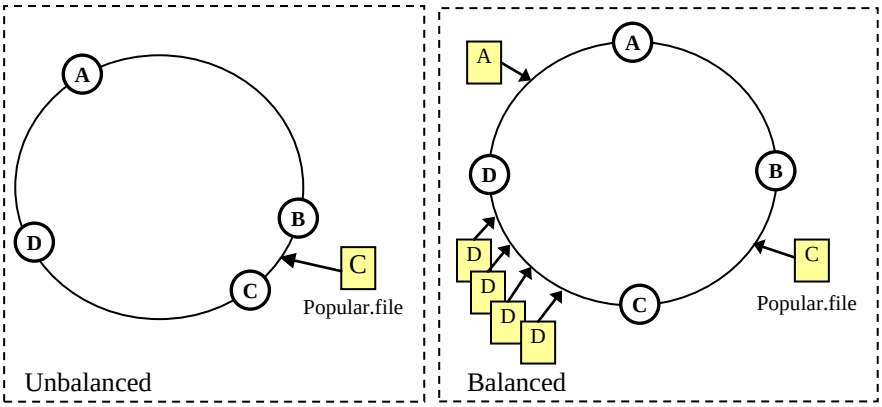

Figure 3.1 displays a address space hash ring before and after balancing where all peer are responsible for the same amount of address space. Even assuming that these peers have the same capabilities in servicing request some crucial balancing issues can still manifest.

A. Address Space Imbalance

Address space misbalancing occurs due to the random hashing nature of DHTs. This type of load misbalancing happens because nodes are allocated random blocs of address space. These address blocs are determined by the placement of nodes on the hash ring described in the Chord background contained in *Section II.* The problem resides in that these nodes are placed randomly, based predominately on what their id, usually their IP address, hashes to using whichever hash algorithm is implemented. Since a node's id is not directly related to their performance or files they contain, it can be said these address blocs are also distributed randomly as well.

While some may argue that since the hash function used is consistent in its operation that given a sufficient number of nodes, the address blocs will eventually be equally distributed amongst nodes [1,5]. But this mandates a certain number of nodes and due to their random nature still may provide weak results. Furthering the problem is the notion of heterogeneous nodes, these node often have different resource capabilities associated with them. Thus a random allocation of address blocs can be seen as a major problem facing DHTs.

Address Space imbalance can be shown in the following example. Take a hash ring, which represents a type of address space in distributed hash tables, and place the nodes described in *Table 3.1* along the ring depicted in *Figure 3.1 Unbalanced* as specified by a hash of their node id.

Node A	56K
Node B	Cable
Node C	T1
Node D	Cable

Table 3.1

Node A	56K
Node B	Cable
Node C	T1
Node D	Cable

Table 3.2

The set of nodes is heterogeneous, each node has an associated bandwidth capacity, and for example Node A is a 56k modem while Node C is a T1 line. Given these constraints some very simple conclusions about peer capabilities can be formed. From bandwidth definitions it can be assumed Node A will be able to handle the least amount of bandwidth. The B and D nodes should be able to handle more than the Node A and about equal to each other. And finally Node C should be able to handle the most of all. But *Figure 3.1 Unbalanced* clearly illustrates an example of address space misbalancing that can happen due to the random operation of a consistent hashing function. This is clearly shown in that a huge disparity separates what Node A and Node C control, in a DHT hash ring a node is responsible for the files whose names hash to positions that

are counter-clockwise to the previous node. For example Node A is responsible for the section between Node A and Node D. Here the problem created by the random addressing shows its face. The nodes are randomly placed based on their node ids not based on how much load they can actually handle thus above we can have Node A, a 56k having a far greater amount of responsibility than Node C a T1 capable host.

Even address space balancing, ideas expressed in *Section II*, do not provide an adequate solution to the issue. As *Figure 3.1 Balanced* shows a perfectly address balanced hash ring where each of the four nodes has the same amount of address space thus is responsible for the same amount of file addresses. But same problems discussed earlier in *Section III* are clearly evident. The equally distributed hashes of the file addresses don't necessarily coincide with equally distributing files. Since files are still placed by hashes of a file's name and not based upon any performance constraints files can end up the responsibility of any node regardless of that node's current load performance. Also to be noted is that this form of address space balancing does not factor in the heterogeneity of nodes. The Node D can be a 56k while Node B can be a T1, which would show a vast waste of resources in terms of Node B's capabilities and severely overburden Node D. Although some recent work in address space balancing has addressed the heterogeneity of nodes [6,7], they still do not address the hotspot problem as the placement of this popular file is still randomized and still can be put on a resource-lacking node.

B. Hot Spots

The second problem in load balancing structured p2p systems is known as the hotspot problem. The hotspot problem occurs when a particular file is fairly popular among users and requested often. Due to the random nature of file placement on the network hash ring, this file can be placed in the responsibility of any node. The consistent hashing function which causes this random file placement gives no indication of popularity or expected load to peers that take responsibility.

Another contributing factor to this problem is the way in which files are mapped onto nodes in typical DHT based protocols. In DHTs, including Chord, files are mapped onto a single node. This is an issue for several reasons. This introduces a single point of failure for a particular file. If a node fails unexpectedly the file risks becoming removed from the network or at the very least incur a heavy overhead to repair the missing node. For a simple example of the hotspot problem assume the nodes in *Figure 3.1* have the attributes listed in *Table 3.2*.

Here in *Figure 3.1* a balanced address space is depicted where the T1 host, Node A, has the majority of the responsibility. This makes it seem as there is no problem but if a popular file is inserted at the unlucky location where Node C is responsible. The problem becomes evident quickly, as even though the addresses are balanced, the resource lacking Node C is in charge of distributing the resource intensive popular file while Node A, a high bandwidth capable host, can sit fairly inactive. The hotspot problem is not only limited to this extreme case but can occur any place a limited amount of resources is in charge of a demanding environment. This can often occur if resources are static or poorly managed between peers.

IV. PROPOSED SOLUTION

The CBDHT protocol can be viewed as a layered overlay protocol at heart. The protocol was implemented as an overlay network over Chord to highlight its ability to be ported to other structured DHT based p2p networks. This overlay layer would be directly called upon by the application layer instead of interfacing with the protocol layer directly.

A. Content-Based Addressing

The idea behind Content-Based Addressing is rooted in the idea of placing nodes on the network based on the files they contain. This typically involves using the information that identifying a resource uniquely, identifiers such as filenames or a MD5 hash of a file's contents.

Addressing must maintain this hash function relationship as it is the cornerstone of how resources are identified in a DHT network. In a similar concept seen in other p2p protocols this could be described as potentially creating virtual nodes at different points in the address space that represent the files contained in the peer. These virtual servers are simply an additional representation of a host in the network and work much like those in address space balancing. The major difference between the virtual servers used in CBDHT versus those created in address space balancing schemes is the methodology behind their placement. In address space balancing placement strategies can vary from random placement to one tuned toward an equal distribution, this bases the need for placement on the needs of the network. Whereas in CBDHT placement is based on the resources of an individual peer. This type of addressing will have the ability to group peers together with similar interests under a single identity.

B. Hierarchical DHTs

The other important theme in CBDHT is the use of a hierarchical DHT. This approach involves layering the network by taking advantage of the hierarchy formed by Content-Based Addressing. The idea of a hierarchical DHT is used to increase the dimensionality of the traditionally flat network scheme of a DHT. The groups are formed by taking like-interested peers and creating a secondary layer, this layer will share the load between members of the group. This will be accomplished by designating a peer "manager" for these groups, called sub networks. This sub network structure could be of any shape, another DHT or even a simple flooding scheme [10]. The importance is that there remains one node that acts as the representative on the initial overlay hash ring. This representative is in charge of routing between other representatives and communication with the sub network. All initial communication between sub networks travels through the top level of the hierarchy.

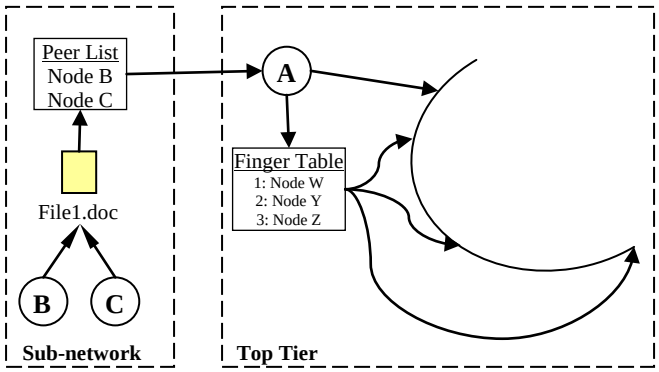

Figure 4.1 shows a close up example of a particular file in a CBDHT network. Here the two layers of CBDHT are clearly shown, Peer A is the responsible node for File1.doc and is in charge handling communication to the sub-network containing Peer B and Peer C .While Peer A acts as a normal Chord node on the Top Tier Overlay.

These sub networks will provide a source of multiple available hosts for a particular resource. These hosts should work as a cohesive unit in order to serve the requests addressed to their location on the tier one overlay. The sub network manager node then takes responsibility of the communication to the underlying group. Using both of these ideas in conjunction can provide an efficient and competitive peer based network capable of self managing peer load while offering redundancy and fairness of file distribution.

C. Sub Networks

Sub networks are the general term used to describe the group of peers formed by the overlap in resources shared by these peers. These groups form as nodes compete for managerial virtual servers on the tier one overlay. The management of the peer relationships within the sub network

is crucial. The behavior of these sub-networks and their respective manager peer will be guided by two main metrics availability and responsibility.

Availability represents the available amount of resources a node has to contribute towards a particular sub network. These resources can be measured any number of ways whether it is the node's maximum throughput, its current load statistics or other user defined metrics. The importance here is that there is a metric installed in order to create competition among nodes for serving files inside the sub network. This competition should favor those nodes with higher resources making them more likely to serve files rather then their weaker counterparts.

The other metric, responsibility dictates which nodes are in charge of representing the sub network on the main tier one overlay. This node's primary responsibilities involve maintaining its presence on this level of the overlay in order to carry out typical routing responsibilities. In the example of Chord this means maintaining a finger table as well as a neighbor set. This manager peer routes incoming and outgoing messages to the member peers within its sub network. Also this node is required to select the most suitable peers within the sub network to service incoming resource requests using availability as a metric. The members of the sub network will occasionally request updates from their tier one representative. These inquires will serve several purposes. First they act as a discovery service for the manager peers allowing them to assign and place the newly discovered node in the sub network. This process will vary based on how the sub-networks are implemented, but usually involve peers competing with one another vying for optimal spots using an availability metric described in *Section IV*. It is then up to the manager node to select a subset of nodes from the subnet to serve the requests presented to it from the overlay ring. This sub-network is an evolving structure much like the tier one overlay. The remaining peers are always in competition with each other in the subnet. As better peers are discovered or evolve the manager node adjusts its selection of the winners to account for this change in peer dynamics.

Also these inquiries allow manager nodes exchange their responsibility commitments if a better candidate is available to manage the sub-network. This is equivalent to attempting to balance the tier one management structure. When these balancing operations are carried out, and the two peers exchange management responsibilities, there will be two choices for the method of exchanging sub-net information. These choices will be based on the how particular subnets are implemented. The leaving manager may provide the context and state of the subnet structure or it may be easier to rebuild the subnet using the discovery process described later in *Sub Section D* using the inquiry/response procedure initiated from subnet peers. This becomes an obvious tradeoff between bandwidth and response time as sending the new manager the state of the subnet uses a certain amount of bandwidth while rebuilding through the discovery method has time overhead in that each node must be discovered individually again.

Another purpose of these inquiries is to serve as a failsafe for manager discovery utility. This is accomplished by the corresponding response or lack of response by a sub network manager. Peers will send out inquiries to discover if a manager exists for a specific file in the network if none exists they will become the manager if they contain the missing resource. Along the same lines if a peer abruptly leaves the network and happens to be manager of a resource, the lack of response will alert a peer from the subnet that they must take over responsibility. Thus this inquiry and response cycle form the baseline communication that is needed to construct and maintain the sub network infrastructure.

The size of these sub networks can be bounded by two main factors. The upper bound is highly dependent on how a particular sub network is implemented. This is often seen as a tradeoff between the amount of state or upkeep that is desired at the manager and the elasticity of response expected. The more state kept locally and maintained incurs higher the maintenance costs but allows for more accurate and reliable response times. The second factor is the more tangible commodity of the number of peers sharing the resource. This number increases from two main sources. Initially files can be seeded and supplemented through the initial set of resources a peer

is willing to share as it joins the network. As more nodes enter that contain the overlapping files the larger a sub-net may become. However as this base can become fairly static and is independent of the heterogeneous nature of file popularity. Issues may arise like the hotspot problem discussed in *Section III*. To combat these forces and increase the sub network peer base, peers upon completion of their request for a particular resource will attempt to join the sub network as a possible provider of that resource. This allows CBDHT to reactively accommodate spikes in popularity by dynamically modifying peer identities within network.

In terms of the infrastructure that forms the sub level in this hierarchical DHT, it is important to note that a particular type or implementation is not required. These sub-networks do not necessarily have to be "networks" at all. They simply have to form a cohesive group that can communicate with their manager responsible for their representation on the first tier DHT. This means a simple list can be utilized just as easily as a full fledged DHT network can be on the underlying level.

D. Network Operations

1. Peer Join

A peer joining a network usually provides some resource or file to share with other peers using the network. From this internal list of resources a peer can construct a list of possible virtual servers based on content based addressing described in *Section IV.A.* At this point in time a peer attempts to join the network under all resources it shares, although it may be beneficial to restrict a peer's footprint on the network to a subset of its shared resources especially when the peer shares a large amount. A resource's address becomes a pseudo-address for the underlying peer. These addresses are created using the methodology in *Section IV.A.* From here peers join similar to hosts joining any other virtual server based network through the use of a bootstrapping process of some kind where a peer already in the network is contacted for neighbor information.

This process identifies the current manger peer of sub network described in *Section IV.C.* This manager then directs the peer on where in the sub network it will reside depending on implementation. The idea of leecher peers, or peers that do not contribute to the network's pool of resources rather only consume resources can be supported by a peer maintaining a certain number of proxy links to the DHT through nodes that are actively participating in the network.

Nodes	Filename	Hash
Node A	File1.doc	A1
Node A, Node B, Node C	File2.doc	A5
Node A, Node B	File3.doc	C2
Node B, Node C	File4.doc	F6
Node C	File5.doc	B3

Table 4.1 a sample set of peers willing to form a network.

In the example shown in *Table 4.1* there are three nodes named A, B, and C. At this time the nodes can be heterogeneous in the nature of their capabilities. The important pieces of information a node must provide to be apart of content-based addressing is simply the hash values associated with each file. This hash value represents a specific space on the hash ring, thus each host can be represented on several places on the ring using this technique. This allows a node to join the network based on the files it is willing to share removing the overhead of responsibility for files it does not initially contain. This also may cause overlap in hosts that contain similar files. A diagram of the hosts in *Table 4.1* can be seen on the hash ring depicted in *Figure 4.2*

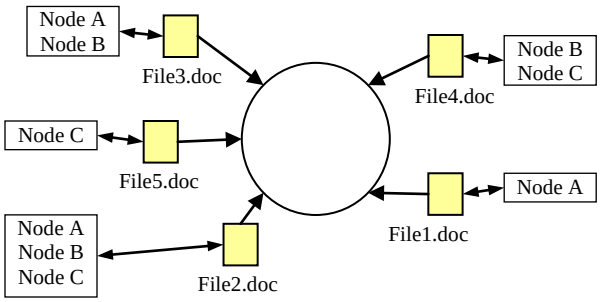

Figure 4.2 Depicts a hash ring constructed from the contents of the nodes described in Table 4.1. These nodes form sub networks amongst themselves with the files they overlap. This creates groups of peers with similar resources to share allowing them to spread the requests for these resources amongst other members of the group.

2. Peer Departure

CBDHT remains resilient to peer departures, both graceful and abrupt. When a node departs the process may be different depending on the circumstances at the time of departure. The main factor in behavioral differences is whether or not the peer is currently a manager of a sub network described in *Section IV.C*. If a peer is a management peer it may choose to exit the network gracefully by choosing and bootstrapping a successor as manager to the sub network. This process is also explained further in *Section IV.C.* The peer may also notify the managers to any sub networks it is apart of and based on the sub networks implementation the manager peer will handle this departure accordingly. But due to the competitive nature of these sub networks this notification may not be necessary as a replacement peer is most likely available.

3. Resource Lookup

Locating a resource on CBDHT is a trivial task as it is very similar to other protocols in the DHT family. Resources are still looked up by the process described in *Section II* for base Chord. The major difference is that since peers join as their file hashes if a file is contained in the network it will have an exact address on the network. Once the request has arrived at the address in the top tier ring, it is now at the current management peer for that resource. It is then up to this manager to further forward and coordinate this request to the optimal sub network peer to handle it. An example of this can be depicted in *Figure 4.2.* In this case a request for File1.doc would be sent to peer A by means of a traditional DHT network. From here peer A must decide whether peer B or peer C is the optimal host to serve the request based on the load operation implemented in that sub network structure. This process is further described in *Section IV.E.*

4. Structure Maintenance

Maintaining the structure of the hierarchical layers and sub network membership is handled by a pair of messages, FILE_INQUIRY requests and FILE_RESPONSE replies. This pair performs the important task of passing information between peers about the current request load at a particular peer and the responsibility associated with a file. This allows peers to make decisions about managing the responsibility of files as well as the sub networks. The effect of this message cycle on a sub network has been discussed in *Section IV.E.*

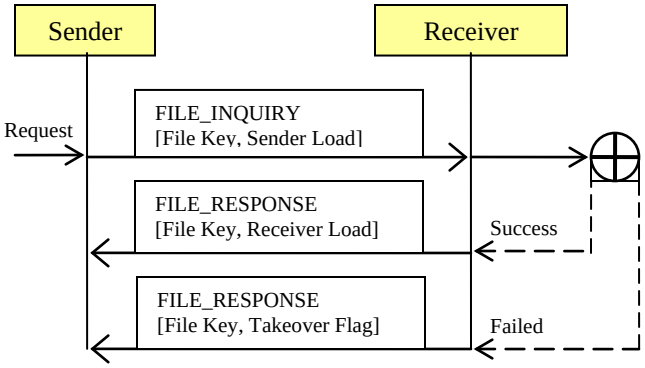

Figure 4.3 describes the maintenance inquiry/response message protocol for CBDHT.

In terms of the top tier overlay, using these messages allows peers the option to give up responsibility if another peer is better suited or under less stress than the receiving management peer. The receiving peer will immediately give up responsibility for that particular resource to the inquiring peer.

E. Load balancing algorithms

It is important to note that load balancing algorithms play a vital role in CBDHT. They are responsible for stabilizing the managers of the first tier overlay hash ring as well as balancing the load between peers on the sub networks. In both sub networks and the first tier overlay, load balancing serves two separate but equally important functions.

By balancing the file distribution of the top level overlay ring, provides a balancing of the routing overhead ensued by maintaining the hierarchical structure as well as what is required to maintain the network on this level. This distribution is reliant on the number of individual resources a peer is sharing as well as the number of peers in the network. This is due to the fact that a virtual server is potentially created for each of these shared resources. Thus it follows that the more resources a peer shares the more maintenance overhead that peer will incur due to the higher number of virtual servers that must be maintained. On the counter side to this the more peers that are in the network the higher chance for overlap in shared resources between nodes. This overlap allows nodes to compete for virtual server spots with one another, effectively lowering the number of virtual servers each individual node must maintain. It is also significant to note that as peers request more resources and begin to share these newly acquired resources the overlap between peers grows driving the number of resources each individual peer must manage even lower. Optimally a balance will be found between the number of peers present in the network and the numbers of resources shared collectively. Additionally the maintenance load can be further balanced based on the heterogeneous capabilities of a node relative to those competing with it to represent the same resource. This would make weaker nodes less likely to be overwhelmed by maintenance upkeep and defer the majority of overlay management to more readily capable nodes. One caveat is that if there is a resource that only a single node contains that node must manage that resource on the first tier overlay ring.

The second area of influence for load balancing algorithms impacts the sub networks. This can be viewed as a type of resource management by the manger of the sub network. Through the manger's direction a subset of peers are chosen from the sub network to handle incoming resource requests for a given amount of time or threshold. The goal of this procedure is to take advantage that a group of peers can all serve as distributors of the resource that the particular sub network is based around. This in turn disperses the load of handling requests over a group of nodes rather than an isolated source as in protocols that have procedures similar to Chord where a single peer is the sole holder of a resource [1]. The flexibility of CBDHT allows for any number

different load balancing schemes to be utilized at the sub network level. Schemes can be based around many metrics, such as number of files served, maximum utilization, bandwidth capacity or many others. Depending on what the protocol would like to stress, a combination of any comparable metrics can be chosen. The importance is that there is a competitive nature to these comparisons that allow the sub network manger to shape its selections to favor unburdened peers. Optimally these load balancing comparisons would reflect the current state and capabilities of peers allowing them to evolve over time as peers become weaker or stronger.

Although the focus of this work is not the comparison of specific load balancing methodologies at the sub network level, it is import to see the multitude and impact of choices available. These resource management techniques may be particularly optimized for the specific sub network implementation chosen. This highlights the flexibility and adaptability of CBDHT, as it allows for several configurations and architectures based on desired behaviors by the network while still maintaining a DHT's fundamental benefits.

V. Experimental Setup

This section will give an overview of how experiments were setup and carried out. Also it will contain implementation notes that give insight into how the protocol described in this work reacts to different conditions. It however is not indicative of the optimal settings for a given application as these should be addressed on a more individual basis. The settings described hereafter are used to highlight the general behavior of the protocol. In addition they will provide a competitive environment to observe performance versus the methodologies outlined in *Section II*.

A. Experimental Setup

The protocol was written in Java 1.4 as required by the simulator PlanetSim. This simulator was chosen due to its abstraction of the underlying DHT protocol. Also being a Java application allowed the implementation to be ported across several environments with ease. As mentioned the simulator was run on several platforms Solaris, Windows XP/Vista with no issues. JVM Size was roughly capped at 1.2GB depending on machine. This amount of memory was required for some of the larger simulations including 250+ peers.

For this work the project PlanetSim [13] was used for all simulations. Each protocol was implemented using this Java based simulator. A data point represents an average of six runs of full simulations done under the specified parameters for that run held in Java property files that describe the various network wide parameters. These simulations were run for a fixed amount of time, which was represented in network steps. These steps can be viewed as a fixed artificial time cycle managed by the network. This way the amount of time was constant across all platforms and protocols independent of background processes. The timed cycles started after the initial network had stabilized, running 100,000 network cycles. This allowed the simulator to take a snapshot of the protocol's statistics without the fluctuation of initial bootstrapping to limit the time frame of data collection. This method captures a view of the network at an arbitrary point in time rather than from time zero.

B. Protocol

The protocols implementation falls under two parts. Protocol processes managed by attributes that can be controlled by parameter passing or configuration handling. Attributes are modified based on the test and protocol being used. *Table 5.1* describes the attributes that were used to modify the actions of the protocol.

Variable Name	Range	Purpose
Load Computation Rate	1 - ∞	The length of time each node takes between computing its network load.
Load Threshold	0.0 -1.0	The percentage by which a competing peer must be superior to the current peer.
Stabilization Rate	1 - ∞	The frequency by which each node performs it inquiry/response cycle.

Table 5.1 protocol attributes

The main functionality of these attributes is modifying the protocol's behavior in dealing with network load. Specifically the attribute named stabilization rate is of significant importance for several reasons. It can be best described as a peer heartbeat. The full cycle of this heartbeat is described in the protocol description in *Section IV.D.* This heartbeat allows nodes to be discovered and share information with one another at the same time. Thus its frequency regulates how often this cycle is completed, and can be seen as tradeoff between network adaptability and peer maintenance bandwidth. The more frequent the cycle the quicker changes will propagate throughout the network, but using a heartbeat also requires more internal maintenance messages to be sent through the network per node increasing the general routing load.

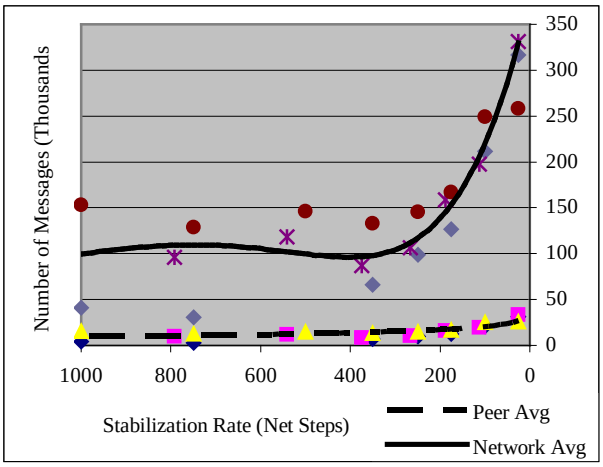

Figure 5.1 shows the effect of the Stabilization Rate on both the average peer's data bandwidth as well as the bandwidth of a constant size (ten peers) network.

Figures 5.1 and 5.2 show the drastic impact the network stabilization rate can have on network behavior. *Figure 5.1* focuses on the bandwidth effect this variable has on each peer as well as the entire network. The individual peers' increase can be roughly viewed as linear in nature although a slight tail is evident as the stabilization rate approaches zero. This however is seen in a much clearer picture if the bandwidth of the network at a whole is observed. A sharp increase shows itself when the stabilization rate moves toward 200. This corresponds with the peers' increase extending beyond linear. An explanation for this behavior is that as a rate of 200 cycles is reached the traffic required to handle these requests begins to exceed 200 cycles at a

particular peer. Therefore the process of handling these requests creates more requests inducing a non linear increase, this problem becomes even more apparent within shorter periods.

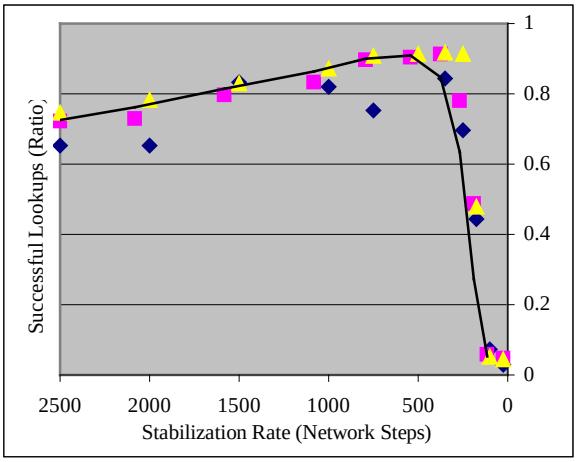

Figure 5.2 illustrates the impact of the Stabilization rate on the Accuracy of resource lookups in the network. Successful inquiries are plotted 0.0 -1 where 1 represents 100% accuracy rate.

The other figure, *Figure 5.2*, describes the effect of network stabilization on the network's accuracy of servicing requests for a specific resource. Here two behaviors can be seen in the graph, a slow gradual increase in accuracy as the stabilization period decreases towards a rate of 500 followed by a sharp drastic decline in accuracy past this point. The gradual increase is explained by peers receiving updated information about which manager nodes are in charge of which resources currently. The longer the period the more likely information is to be stale and lead to miscommunication. On the other hand as the stabilization rate approaches zero a viscous drop off in accuracy is seen. This can be explained in conjunction with *Figure 5.1* where around this period, a significant increase in overall network traffic is observed. This may cause bottleneck conditions at peers in turn causing network congestion and message loss at extremes. But what may weigh even more heavily is the actual movement of peers in sub groups themselves. Within this shorter period there lies a greater probability for peers to move within the sub network while a request is in transit towards the management peer. This can be categorized as a localized thrashing problem at the sub-network level. This problem is further exacerbated by the fact that in this specific implementation the management peer also represents the servicing peer as well. This will cause more movement on the tier one DHT than other implementations of CBDHT that separate this responsibility.

The remaining two variables deal directly with how a peer's load is evaluated by another peer. The load threshold is simply a percentage used to compare two competing nodes against one another. The established node is always at advantage in comparing load while the challenger node must overcome the load threshold to be considered superior. The lower the percentage the less likely peers are to give up their positions as it requires significantly greater performance to be considered better. The bottom a percentage of 0% would cause peers to never consider a competitor peer superior. This would create a static network where nodes do not move based on load and rely on network churn for movement. The effect on peer load is illustrated in *Figure 5.3*. Here the relatively high loaded peers are depicted on the low end of the load threshold. Conversely the higher the percentage the more likely a peer swap will occur. The upper limit to is 100% or a load threshold of 1.0, allows peers to be evaluated as equals. This is also evident in *Figure 5.3* where the higher load threshold causes peers to switch and distribute load more

frequently contains a much lower load distribution. This can create significant thrashing however as peers will swap with the slightest fluctuation of load.

The load computation rate, much like the file stabilization rate, is a frequency. This frequency's job is to regulate how often a peer computes its internal load. The actual process and metrics used in this load computation can change significantly based on implementation. The load computation process can have varying effects on the overall behavior of the protocol. This is especially true in load computations that involve degradation of load over time. However in a more static context it represents an accuracy of load measurement at any given time. Load computation rates that have long periods may incur peer lag in terms of accurately representing that peer's current load. Although computation rates that apply themselves in a more frequent manner may find themselves more susceptible to peak load spikes. Thus the load computation rate can be seen as an instrument to smooth a node's load value over a portion of time.

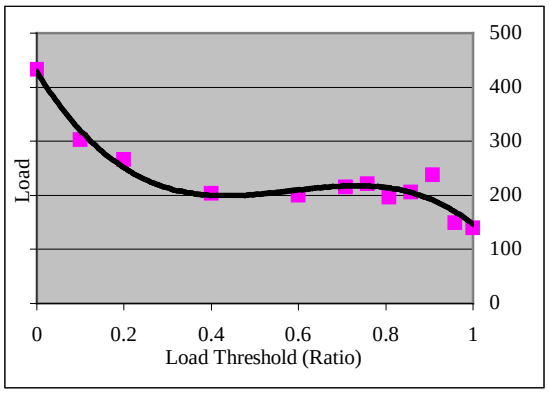

Figure 5.3 shows how the average load of a peer is related to the load threshold. Load of a peer is measured in the number of requests received over the course of its time in the network.

The load computation used in this work is a simple request count for a given peer. While this is a fairly trivial way to look at load it will still give insight into how CBDHT distributes incoming requests.

As discussed in *Section IV.C* the sub network structure that underlies the tier one DHT plays an integral part in the behavior of the protocol. Specifically the structure of a sub network has a direct correlation to the selection of competitor peers to serve as distributors for an individual resource. There exists any number of methods to implement these structures; implementations can be as complex as a full DHT sub network or simply a list of available peers maintained by the resource manager.

The sub network implementation for this work can be described as a transparent list of peers competing for the current serving peer's position in the network. For this work the current management peer also accepts the responsibility of serving requests for the particular resource it is managing. As peers prove to be significantly superior for handling the load presented to the current management peer this node will take over both the routing responsibilities on the first tier DHT as well as the request handling responsibilities for that resource.

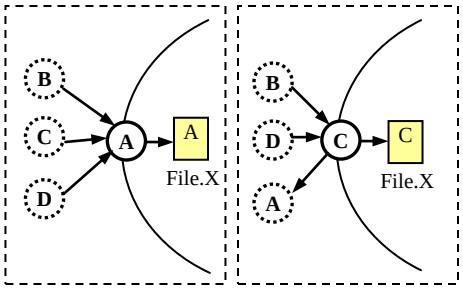

Figure 5.4 shows a typical sub network in this implementation of CBDHT. Peer A is the manager of this sub group and thus responsible for servicing requests for File.X. Meanwhile Peers B,C,D which also contain File.X will periodically compare their internal load versus the current manager peer A. In the next panel, Peer C has been deemed superior host compared to Peer A. Peer C will now handle the management of File.X. Peer A will now join the transparent sub network with Peers B and D.

The importance of load balancing algorithms to CBDHT was highlighted in *Section IV.E*. As described the load metric for this work is based off of a simple count of the number of requests received by that node. Thus the load balancing algorithm for this particular work is computed by:

$$L_E \times \text{Load Threshold} \geq L_C$$

Where L_E represents the load of the established peer being queried and L_C is the challenger peer attempting to takeover the distribution responsibilities. Due to the implementation described in sub network structure the load of routing and servicing requests uses the same algorithm. This was used since both tasks are serviced by the same peer. Routing maintenance load and resource service load are balanced using the same operation. No state is required to be transferred between management peers as the current implementation relies heavily on the inquiry-response cycle described in *Section IV.D*. An example of the load balancing transaction used in this implementation can be viewed in *Figure 5.4*. It illustrates that the incoming superior peer now takes both routing and servicing responsibility as a member of the tier one overlay in the same position that the departing peer resided in.

The specifications highlighted in *Table 5.2* describe how CBDHT will be implemented in this work. This instantiation will be used to compare CBDHT versus other DHT instantiations in a quantitative analysis similar to the procedures followed in this section. Again these specifications do not promise to hold the most optimal settings but rather a generalized solution for this specific implementation.

Variable Name	Value
Load Computation Rate	500
Load Threshold	0.85
Stabilization Rate	500

Table 5.2 describes the parameters chosen for this implementation of a CBDHT network.

C. Metrics

Metrics of particular interest in the simulations can be described in three major categories, routing performance, peer efficiency and network efficiency. Routing performance is the most straight forward metric in this set. It is a simple measure of how efficiently a message can get from Peer A to Peer B on the network. This metric can be measured in several ways. These can be time based, as in lag time, or node based, as in the number of nodes a message must traverse. In these simulations routing performance was measured on a per hop basis. As a general attribute DHT routing performance remains fairly good a log(n) where n is the number of nodes in the

network.[1] But increasing this efficiency can be beneficial in response times as well as a reduction of routing traffic received per node network wide. Thus even though generally log(n) time will be observed it is still possible to identify which protocol may operate better consistently in relation to others.

Peer and network efficiency make up the second grouping of metrics. These metrics are correlated as one generally can be a product of the other. The more individual peers are burdened, the more the network or at least significant portion of it will also be burdened. This holds in the opposite case as well, where when peers are generally under-burdened the network load remains light as well. This however does not take in consideration peers as individuals or how one group of peers can relate to another in performance. This can lead to cases where network efficiency is good but a select group of peers are being exploited to achieve this result, and thus their individual efficiency will suffer. So it is important to separate these two entities when looking to compare performance based results in these protocol implementations. The goal of CBDHT is to provide a fair distributive burden on individual nodes. The definition of fair in CBDHT takes into account a peer's ability in relation to the others in its sub network group. This will allow peers to share their load over a given number of peers based on how the load balancing algorithms are implemented, for this work a description is available earlier in this section.

VI. RESULTS

A. Problem Description Example

The initial network that was simulated was described in *Figure 3.1*. This network was selected to illustrate some of the problems that frequently face DHT based networks and how typical approaches to solve the issues of load balancing could fall short. The simulation procedures described in *Section V* were carried out on this network in a similar fashion to the experimental setup testing described in the same section. The implementation of this network focused on reenacting a simple case of the hotspot problem where one node unfortunately holds a particularly popular file within its address space. This file would receive the majority of the total requests sent by the other four peers and thus expected to receive the majority of the load on the network.

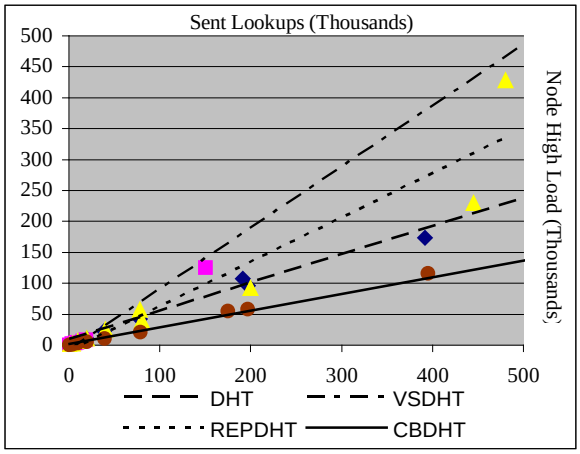

Figure 6.1 depicts the highest load for a peer in the sample network described in Section I. CBDHT is shown by the circle points containing the lowest burdened peer while a virtual server network (square points) contains the peer with the greatest burden.

The results highlighted in *Figure 6.1* and *Figure 6.2* confirms the problems viewed in *Section II*. By looking at the highest load achieved by an individual peer in the network the effect of the

hotspot problem can be identified. This is because the peer that contains the selected popular file should in turn be burdened with the most request based load on the network. This is clearly observed in the data in *Figure 6.1*, by looking at the four protocols presented for comparison. The baseline protocol DHT exhibits the highest load for a single peer, this load increases along with rising requests. This was expected as under a DHT based network, a file's responsible peer does not change under various load conditions; this causes whichever peer that initially gains responsibility for this highly sought file to undertake all requests for this resource. This peer has no way to redistribute this load, save for leaving and rejoining the network under a new hash id and thus a new location in the network. A virtual server derived DHT fares little better than plain Chord in this example; this is because a virtual server DHT attempts to distribute the load by better distributing address space. This does not work for the hotspot problem as a hotspot is represented in a DHT by an individual location on the network ring, this means no matter how much the address space for the network is divided it cannot divide this element into multiple locations to be handled by multiple peers. This is mainly because both plain DHT and virtual server strategies distribute load based upon the distribution of multiple files rather than distributing the actual load incurred by any particular file. This methodology differs from protocols using replication strategies and the CBDHT protocol as these methods deal directly with the distribution of load for a specific resource rather than a vague un-quantified collection of files. Thus as expected a DHT that uses a replication strategy does in fact have a lower max peer load than the previous protocols. However it does not have a lower load than CBDHT, this is mainly because of one fundamental issue with replication strategies. Replication strategies are only effective if they are included into frequented routing paths for the popular replicated file. This causes replication to have a load bias for those nodes in the routing path, which in the case of DHTs are typically peers that have hashes that come before the desired resource in a counter clockwise fashion around the address ring as described in the Chord background of *Section II*. The closer a peer is to a desired resource the more likely this peer will be chosen as a replication spot as well as being more likely to actually serve the request once a replication has been placed on it versus a peer that resides further from the desired resource. CBDHT however does not carry this bias thus it has the ability to distribute regardless of a specific peers hash placement. This allows CBDHT to achieve the performance depicted by spreading the load out as equally as possible between interested nodes.

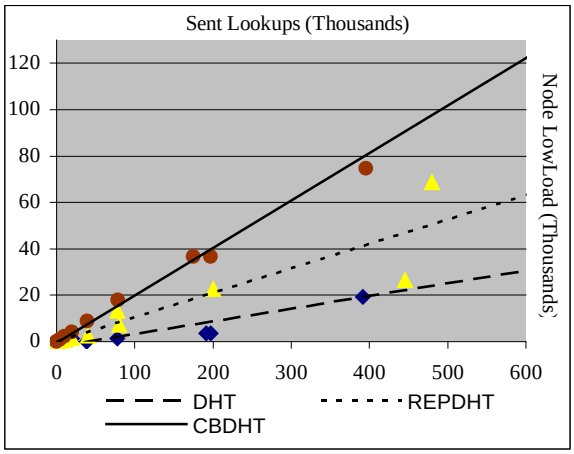

Figure 6.2 shows the lowest load in the network for a peer in the sample network described in Section I. CBDHT is shown by the circle points containing the highest peer. This shows CBDHT spreads the load equally amongst all interested peers causing the noted increase.

CBDHT uses all nodes interested in sharing a given resource to distribute load. Particularly in this implementation all nodes contained in the sub network for a particular file are eligible to

handle requests for the file (in other implementations this coverage may be reduced or modified). *Figure 6.2* shows the lowest load recorded by a node in the network. As observed, plain DHT, replication, and virtual servers all follow similar patterns where even in this small network of four peers a node can escape a majority of the burden while reaping the benefits of the network. CBDHT as stated before includes all members in the sub network thus the lowest load on the network is significantly higher than the other protocols for the same number of requests. The load of the lowest burdened peer can be viewed as linear to the amount of requests.

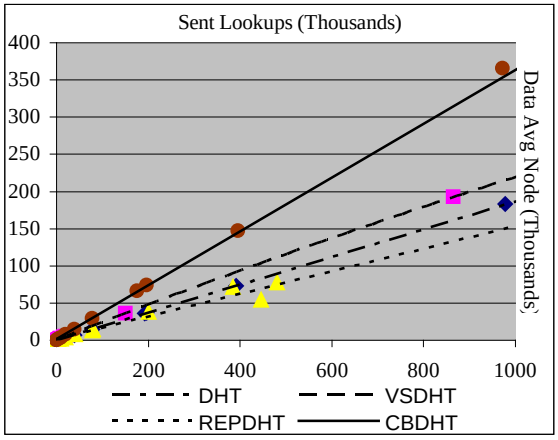

Figure 6.3 shows the amount of data traffic passing through the average node in a simulated network. This traffic includes file requests as well as any maintenance messages required by the protocol for its operation. The higher the amount versus plain DHT (diamond points) the more the protocol costs in upkeep costs. As shown CBDHT (circle points) costs marginally more than the other protocols but these costs are still linear nature .

While lowering the load of the highest strained peer is in effect the goal for load balancing, this goal must be valued by not only how much a particular methodology gains but also by how much a methodology costs. A perfectly balanced network is not desired if the cost of maintaining this balance is detrimental to the efficiency of its operation. This cost is represented by maintenance bandwidth. Maintenance bandwidth can take many forms in different protocols for example the cost of adding a virtual server or the cost of sending a status message to another node. The easiest way to quantify this is to measure the total network traffic while doing a specific and repeatable task on a base case and then running the same repeatable task for the experimental case. The added amount of network traffic between the two protocols would represent the cost of employing a particular protocol over another. The graph in *Figure 6.3* shows exactly this. The repeatable task is the number of requests sent for resources network wide and the network traffic is the summarization of all messages received by peers. The results are as expected with DHT representing the baseline amount of network traffic. Replication based networks have a similar footprint as the baseline DHT. This is because this replication strategy simply observes network traffic internally at the peer level, the majority of replication costs lay in another domain, forced resource transfers, to be discussed in *Section VI.D.* Virtual server based networks fair slightly worse as the cost of entering and maintaining multiple ghost representations is expensive in upfront costs as well as continual maintenance of these locations. While CBDHT does have costs tied to it, its cost increase is fairly linear versus an increase in load. This maintenance cost is directly tied to the cycle discussed in *Section IV.D,* in which peers communicate with one another about their relative load to one another. So a decrease in overhead

is available at the cost of the frequency at which peers can switch responsibility due to load burdening.

This sample network is a limited example that serves its purpose of highlighting issues with the hotspot problem however it does not give the full picture of how these protocols would act in typical network situations. Most notably this network is static, where the set of nodes is constant through out the simulation's lifetime. This is important as the joining and leaving of nodes effects each protocol in different ways, particularly in how each strategy responds to resource misplacement. Also this simulation only contained a small amount of peers and shared resources this makes it difficult to observe how each protocol's methodology scales when dealing with a large peer base as well as an increase in resources to manage.

B. Typical Networks

Networks were chosen at various sizes 10 peers, 40 peers, and 400 peers to represent an array of typical sizes and derive a sense of scale from the protocols. Scale remains one of DHT's central strengths and it is important that any proposed improvements maintain this scalability in order to remain viable.

Networks were compared by the ratio of their highest load burden peer during the run and the average optimal peer load. The average peer load represents the optimal load of any node in the network. In a network where all node nodes are treated equally capable, when the highest load is equal to the average load this represents a perfectly balanced network where all peers handle their fair share of the request load. The wider this gap and more frequent the number of deviations from the average peer load the more drastic the imbalance between peer loads are. In a heterogeneous network if the protocol accounts for peer capabilities, the highest peer load may deviate from the average peer load by a given amount that represents this peer's superior capabilities over the average peer. *Figure 6.4* depicts this metric as a function of increasing network load in the form of additional resource requests. The important fact to take away from this figure besides the numerical values is the shape of the curves for each protocol.

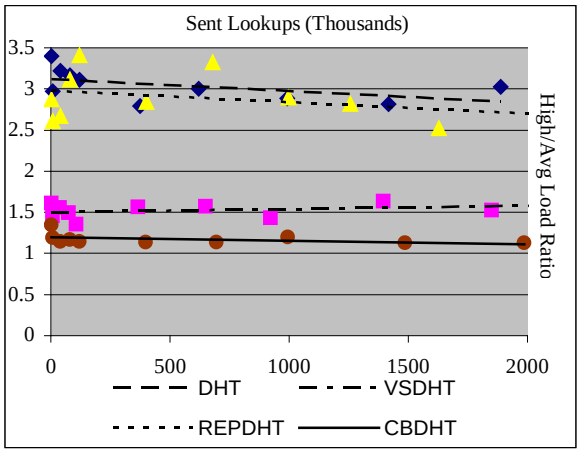

Figure 6.4 depicts the high load and the average load in a ratio against the amount of sent lookups. This in essence show how effective a protocol is in distributing load across its peers. The closer to one the more efficient a protocol is, CBDHT is easily the most efficient (circle point) while a virtual server network (square points) remains second.

Each protocol's data exhibits generally a straight line when plotted against rising network load. This means even as network stress is rising all protocols remain constant in their ability to distribute load. The results show plain vanilla DHT fairs the worst as expected since DHT has no built in protection against the hotspot problem any peer that carries a popular file will become the

highest burdened peer. Replication based protocol follows which is most likely due to the small network size. This causes most routing paths in a DHT network to be small many of which routes falling to a single hop. This considerably reduces the number of viable peers to make file replications on and significantly decreases the effectiveness of this load balancing technique on smaller-medium sized networks. Virtual Server based load balancing does significantly better although it can not remove the hotspot issue it effectively mitigates load based on the distribution of requests for several files. CBDHT does considerably better than the other protocols its high to average ratio hovers around 1.15 or roughly 86% of the optimum value. This value of 86% falls in line well with the threshold value of 85% chosen in the experimental setup in *Section V* for the required amount of load over the current peer to take over responsibility. This shows that CBDHT is clearly distributing the load equally among all available peers in the network regardless of individual file popularity. *Table 6.1* shows that these results remain static through a variety of network configurations.

	DHT	RepDHT	VSDHT	CBDHT
Small(S)	3.58	2.99	2.73	1.17
Small(M)	2.92	2.80	1.48	1.22
Medium(S)	3.04	2.90	1.52	1.17
Medium(M)	3.95	3.78	1.78	1.18
Large(S)	48.64	33.64	45.80	11.34

Table 6.1 shows the high to average load ratio for different network sizes and peer contributions. The format is Network Size (Amount of peer files). Small networks were 10 peers, medium 40 peers and large 400 peers. Small peer resource contributions were between 5-20 files per peer while medium contribution ranged from 75-150 per peer.

In *Table 6.1*, it is evident that for most small to medium size networks the values of high to average load ratio stay similar in nature. This consistency breaks down as the simulations move up to the next level of peers in the network. This is caused by a significant increase in file requests from the additional peers. This is especially evident in the non CBDHT protocols as the majority of their ratios grow rapidly. CBDHT is also affected by the increased amount of requests however at a reduced level. This is caused by the increased amount of requests that can be received in between the inquiry-response cycle described in *Section IV.D*. This allows a node to accrue a higher than average amount of load in a short amount of time. This can be reduced by lowering the time in between node switches through the inquiry-response cycle or using a different more responsive load balancing protocols at the sub network level.

C. Network Churn

Networks that are not static, in which peers may join or leave as they wish, introduce yet another important environment. A network built around a dynamic peer base must account for these nodes' departure in terms of resource allocation and routing paths. In simple DHT schemes node departures may be expensive as basic DHT operate under the many files to one node scheme clarified in Section II. This scheme places the burden on the successor of the departed peer to become responsible for those files that were contained by the peer that left. This shift in responsibility occurs regardless of the peer's desire to contain the files or resources abandoned. It must locate these resources and transfer possession of them from other peers if available. CBDHT as described in *Section IV* has built in redundancy through the existence of sub-networks containing peers that contain the resource readily available to become responsible for it. This enables CBDHT to continue without much additional cost even in high churn environments depicted in *Table 6.2*.

	DHT	RepDHT	VSDHT	CBDHT
Constant	5.32	4.38	3.05	2.58
Dynamic	6.29	5.26	3.39	3.21

Table 6.2 displays the results similar to Table 6.1 but this time highlighting non-static networks. Two classes of dynamic networks were measured, networks with a constant amount of peers entering and leaving and networks where the churn is accelerating. In the constant networks an average of 400-500 peers join and leave the network while increasing the amount of requests processed. Dynamic churn networks the number of transient peers was between 300 and 1200 respectively with a constant amount of requests.

D. Other Benefits

As important as benefits quantified by metrics are in terms of comparing protocol methodologies, it is also important to observe the differences in these networks on a macro level. These types of comparisons usually compare and contrast structural methodologies of the protocols rather than data quantified by metrics. While not necessarily dealing with load balancing directly the following highlights some of the features CBDHT can hold over other DHT implementations.

CBDHT offers the opportunity for a network to form independent from any centralized/distributed resource manager. This allows CBDHT to escape the overhead and burden of having a resource manager place resources on peers in strategic locations regardless of an individual node's benefit. CBDHT can be started with any given set of peers that are prepared to start a network. The distributive nature of the resource and peer management is built into the operation of the protocol. This resource management is based on the performance and capabilities of individual peers. This information is acquired by peers through a sequence of messages highlighted in *Section IV.D.*

Another benefit to the CBDHT protocol lies in how it is structurally viewed as a hierarchical DHT, this allows several nodes to reside in the same virtual space. This enables CBDHT to break the "many file"-to-"one node" relationship that exists on Chord and other DHTs. The execution of this break is important as it forms nodes into proactive redundant resource managers. This contrasts with the slower more costly reactive approaches that first requires a peer to identify the missing resource and then locate a copy on the network. CBDHT's competitive sub-network algorithms enable a quick recovery and remain resilient from even ungraceful peer departures from the network as peers with similar resources will step in to fill the void. This behavior helps both in the management of the sub networks themselves as well as with the actual distribution of requested resources. This also beneficially affects peers joining as well. Initially when a node joins it must find out for which files it is responsible for as its allocated responsibility is randomly assigned as a part of its random hash address. As it locates these files it must locate replications on the network and retrieve them or route all requests for the resource to the peer that holds them. This is considered additional upkeep because this joining peer may or may not have an interest at all in this resource but is now responsible for its location and distribution, whereas in CBDHT only peers that are interested or already contain this resource will handle its responsibility.

CBDHT also offers a logical way of improving the routing paths of messages between peers. It accomplishes this by using the fact that peers can represent themselves in multiple locations on the top tier DHT as sub network managers or members of a sub network group. A peer then has its choice as to which location it would like to originate the correspondence from. By starting at the location with the hash that has the closest proximity to the destination, this peer can reduce to number of subsequent hops seen by the message. While this may not reduce the order in which a

peer can locate its destination, O(logN), it can beneficially reduce the magnitude of this procedure.

VII. Conclusion

While the general benefits and accomplishments of CBDHTs were described in this work, these benefits can be further extended and customized on the basis of their implementing application. Work on fully describing and discovering this extensibility should be done. In order to fully optimize the benefits achieved through the use of CBDHT an in depth study on the effects and application of several sub network and load balancing operations should be carried out. By optimizing these portions of the protocol it can be hoped to further reduce the maintenance costs associated with the network.

This paper showed that content based addressing can significantly improve the fairness between peers in terms of load balancing requests for the resources served by the network. CBDHT exhibits resilience to different network sizes and load conditions even with the naïve load balancing schemes presented. These schemes are open to a high amount of flexibility in their implementation allowing for the protocol to be further developed to tailor to its specific application. The focus has been on the methodology of addressing peers by their individual content and balancing based on their current load and capabilities. This allows DHTs to identify peers by more qualitative means instead of a random hashing function. Presenting a CBDHT based network with a high level of fairness and resiliency while maintaining the key benefits of the core DHT implementation.

VIII. References

[1] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker. A Scalable Content-Addressable Network. In *Proceedings of SIGCOMM 2001*

[3] A. Rao, K. Laskshmunarayanan, S. Surana, R. Karp, I Stoica. Load Balancing in Structured P2P Systems. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), 2003.*

[4] D. Novak. Load Balancing in Peer-to-peer Data Networks. In *MEMICS 2006, 2nd Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*

[5] D.R. Karger, M. Ruhl. Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems. In *ACM Symposium on Parallelism in Algorithms and Architectures, June 2004.*

[6] G. Giakkoupis, V. Hadzilacos. A Scheme for Load Balancing in Heterogenous Distributed Hash Tables. In *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing, July 17-20, 2005, Las Vegas, NV, USA*

[7] P.B. Godfrey I. Stoica. Heterogeneity and Load Balance in Distributed Hash Tables. *In INFOCOM 2005.*

[8] Y. Xia, A. Dobra. Ideal Load Balancing Techniques in Structured Peer-to-peer Networks. *Submitted*

[9] Z. Xu, R. Min, Y. Hu. HIERAS: A DHT Based Hierarchical P2P Routing Algorithm. In *Proceedings 2003 International Conference on Parallel Processing*

[10] L. Garces-Erice, E.W. Biersack, P.A. Felber, K.W.w Ross, G. Urvoy-Keller. Hierachial Peer-to-peer Systems. In *Proceedings of ACM/IFIP International Conference on Parallel and Distributed Computing (Euro-Par)*

[11] K. Kenthapadi , G.S. Manku. Decentralized Algorithms using both Local and Random Probes for P2P Load Balancing. In *Proc. of 17th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA) 2005*

[12] PlanetLab. http://www.planet-lab.org
[13] PlanetSim. http://planet.urv.es/trac/planetsim/
[14] Napster. http://www.napster.com
[15] A. Rowstron, A-M. Kermarrec, M. Castro and P. Druschel. SCRIBE: The design of a large-scale event notification infrastructure. *NGC2001, UCL, London, November 2001*
[16] Bassam A. Alqaralleh, Chen Wang, Bing Bing Zhou, and Albert Y. Zomaya. A Proactive Method for Content Distribution in a Data Indexed DHT Overlay. *In the Proc. of the 3rd International Conference on High Performance Computing and Communications (HPCC 2007), Houston, USA*